日本ソフトウェア科学会第 27 回大会 (2010 年度) 講演論文集

Fine Grain Task Manager Cerium のチューニング

金城 裕　　河野 真治

現在 Cell/PS3 または Mac OS X 上で動作する Fine Grain Task Manager である Ceirum を開発中である。Cerium Task Manager は、Cell/PS3 または Mac OS X 上で動作する Open CL 的な Fine Grain Task Manager である。ソフトウェアレンダリングエンジンと Word count を例題として、Task Manager の実装時の問題を洗い出している。メインメモリ上の Task を各 Core が受け取る際や、その終了を通知する際に待ち時間が生じる。その待ち時間を削減するために、Task array を提案し実装した。その効果について報告する。

1 概要

CPU の処理速度の向上ためのクロック周波数の増加は、発熱や消費電力の増大により難しくなっている。そのため、クロック周波数を上げる代わりに、CPU コア数を増やす傾向になった。マルチコアな CPU の性能を発揮するには、処理をできるだけ並列化しなければならない。それはアムダールの法則により、並列化できない部分が並列化による性能向上を制限することから言える。つまり処理速度の性能向上は、ハードウェアだけでなく、ソフトウェアを並列処理に適したように実装することにもかかっている。そのためにはプログラミングの支援をするフレームワークが必要になってくる。そこで Fine Grain Task Manager である Ceirum を開発中である。現在 Cerium は、マルチコア CPU の例題として Cell に対応している。また、支援するプログラミングの対象の 1 つとしてゲームを選択し、PS3,Mac OS X 上でのゲームフレームワークとしても動作する。その Cerium のチューニングをするうちに、各 Core において、割り当てられた Task が終わり、次の Task を待つ時間が処理速度を遅くしていることがわかった。そこで待ち時間を削減するために、各 Task 生成方法、スケジュールリング方法の変更、複数の Task をまとめて扱う TaskArray を提案し実装した。その効果について報告する。

2 Cell Broadband Engine

Cell Broadband Engine は、ソニー・コンピュータエンタテインメント、ソニー、IBM, 東芝によって開発されたマルチコア CPU である。Cell は、1 基の制御系プロセッサコア (PPE:PowerPc Processor ELement) と 8 基の演算系プロセッサコア (SPE:Synergistic Processor Element) で構成される。各プロセッサコアは、EIB (Element Interconnect Bus) と呼ばれる高速なバスで接続されている。また、EIB はメインメモリや外部入出力デバイスとも接続されていて、各プロセッサコアは EIB を経由してデータアクセスをおこなう。

この PPE と SPE の 2 種類の CPU を、プログラマ自身が用途に合わせて適切に使い分けるように考慮する必要がある。

2.1 PPE

PPE は Cell BroadbandEngine のメインプロセッサで、複数の SPE をコアプロセッサとして使用することができる汎用プロセッサである。メインメモリや

Tuning of Fine Grain Task Manager Cerium

Yutaka Kinjyo, Shinij KONO, 琉球大学大学院理工学研究科情報工学専攻並列信頼研, Dept.Concurrency Reliance Laboratory, Information Engineering Course, Faculty of Engineering Graduate School of Engineering and Science, University of the Ryukyus.

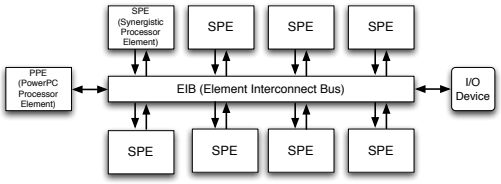

図 1 Cell Broadband Engine Architecture

外部デバイスへの入出力、SPE を制御する役割を担っている。PPU(PowerPCProcessorUnit) は、PPE の演算処理を行うユニットで、PowerPC アーキテクチャをベースとした命令セットを持つ。

2.2 SPE

SPE には 256KB の Local Store(LS) と呼ばれる、SPE から唯一、直接参照できるメモリ領域があり、バスに負担をかける事無く並列に計算を進めることが出来る。SPE からメインメモリへは、直接アクセスすることは出来ず、SPE を構成する一つである MFC (MemoryFlowController) へ、チャネルを介して DMA(DirectMemoryAccess) 命令を送ることで行われる

2.3 DMA

SPE は LS 以外のメモリに直接アクセスすることができず、PPE が利用するメインメモリ上のデータにアクセスするには DMA を用いる。DMA(Direct MemoryAccess) 転送とは、CPU を介さずに周辺装置とメモリとの間でデータ転送ことで、Cell の場合はメインメモリと LS 間でデータの転送を行う。手順としては以下の様になる。

1. SPE プログラムが MFC(MemoryFlowController) に対して DMA 転送命令を発行
2. MFC が DMAController を介して DMA 転送を開始。この間、SPE プログラムは停止しない。
3. DMA 転送の終了を待つ場合、SPE プログラム内で転送の完了を待つ

2.4 Mailbox

Mailbox とは PPE と SPE 間の 32 ビットメッセージの交換に用いられる。Mailbox では 3 つの振る舞いが出来る様に設計されている。

1. SPU Inbound Mailbox
   PPE から SPE へデータを渡すためのキュー。キューのエントリ数は実装依存による が、研究環境では最大 4 個までのデータを蓄積できる。このキューが空の場合は、SPE は、データがメールボックスに書き込まれるまでは、命令でストールする。読み出すデータの順番は書き込んだ順番に保証されている。
2. SPU Outbound Mailbox
   SPE から PPE へのデータを渡すためのキュー。研究環境では最大 1 個までしかデータが蓄積できない。
3. SPU Outbound interrupt Mailbox
   SPU Outbound Mailbox とほとんど同じだが、このキューでは SPE からキューにデータが書き込まれると、PPE に対して割り込みイベントが発生し、データの読み出しタイミングを通知する事が出来る。

3 Cerium とは

Cerium は TaskManager、レンダリングエンジンと SceneGrpah の 3 つの要素から構成されており、PS3、Mac OS X、Linux 上でゲームフレームワークとして動作する。ゲーム中のオブジャクトの振る舞いやルールは SceneGraph で管理し、それらの動きやレンダリングの処理を動的に SPE に割り振るカーネルとして、TaskMnager が用いられる。PS3 の Graphics Engine の仕様は公開されておらず、ソフトウェアレンダリングエンジンを実装する必要があった。

3.1 TaskManager

TaskManager は、Task と呼ばれる、分割された各プログラムを管理する。Task の単位はサブルーチンまたは関数とし、Task 同士の依存関係を考慮し、実行可能状態になった Task を各 SPE に割り振る。Task

は通常 PPE で生成され、SPE に送られる。SPE では、受け取った Task をパイプラインに沿ってステージを遷移しながら複数の Task を同時に実行していく。

4 Cerium における Task

Task は TaskManager を使って生成する。Task を生成する際に以下のような要素が設定可能である。

1. input data
2. output data
3. paramater
4. cpu type
5. dependency

input,output data, paramater は関数でいうところの引数にあたいする。cpu type は Task が PPE, または 6 基ある SPE のどれかで実行されるかを示すもの。dependency は他の Task との依存関係を示す。依存関係の情報は PPE 側が持っている。SPE から PPE へ実行し終わった Task が通知され、PPE 側ではその通知に沿って依存関係を処理していく。例えば、Task A が Task B の実行完了をまって、実行可能状態になるとする。はじめ Task B はどの Task も待つ必要がないので、SPE に送られ、実行される。Task B の実行が完了すると、SPE から Task B の完了通知が Mail で PPE 側へ通知される。Task B が実行完了の通知がきたので、Task A の待ち Task から、Task B がはずされ Task A は SPE に送られる。以上のように SPE からの Mail を使った通知によって、Task の依存関係を解決している。待ち Task を持っている Task は wait queue と呼ばれるキューに、待ち Task のない Task は active queue と呼ばれるキューに入れられる。active queue にある Task を SPE に送る。()

4.1 Task のスケジューリング

SPE は、Task を一つずつ受け取るのではなく、ある程度まとめて受け取る。それを TaskList と呼んでいる。TaskList に沿って Task を実行していき、Task 毎に実行完了の Mail を送る。TaskList の Task を全て実行すると、次の TaskList を要求する Mail を PPE 側に送る。

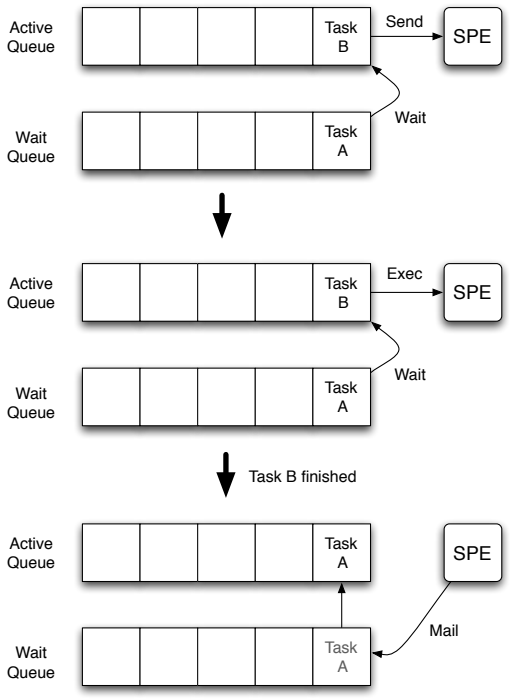

図 2 依存関係の解決

5 TaskArray

WordCount の場合, 対象となるファイルによって大量の Task が生成される。レンダリングエンジンの場合、PPE 側で実行すべき Task がある。その時に、PPE が自分の Task を完了し、Task の依存関係を解決するのに時間がかかってしまう。そうなると SPE からの TaskList 要求への反応が送れ、SPE の待ち時間が生じるはずである。それを解決するために TaskArray を提案、実装した。TaskArray は、複数の Task をまとめて扱うことができる。それによって依存関係を解決すべき Task の数が減り、解決の手間が省け、SPE からの TaskList 要求に応答しやすくなる。また、一度に TaskList に多くの Task を登録できるため、SPE 側からの TaskList 要求の回数が減り、待ち時間が生じる可能性が減る。さらに TaskArray の長さによっては、パイプラインが長くなり、DMA の転送時間を多く隠すことができる。WordCount とレンダリングエンジンのおいて、Task を TaskArray

にし、効果を検証した。

5.1 WordCount の Task

まずは例題の WordCount の Task について説明する。WordCount の Task は以下の二つである。

1. WordCountTask
2. PrintTask

WordCountTask は、input で与えられた data を word count し、output data に書き出す Task である。PrintTask はすべての WordCountTask の実行完了を待ち、output へ書き出された値を集計し出力する Task である。一度に SPE に渡せるデータ容量は DMA の仕様上 16Kbyte までである。さらに転送する際には 16 の倍数 byte である必要がある。

5.2 WordCount の Task 設定

wc する file をメモリへマッピングし、WordCount-Task の input に、file data のアドレスを 16kbyte ごとに指定していく。

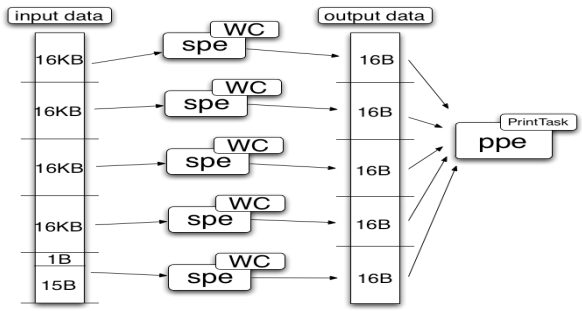

図 3 WordCount における Task の流れ

PrintTask は WordCountTask を待ち Task と設定し、WordCount がすべて終わらないと、実行されない。

5.3 WordCountTask の TaskArray 化

WordCountTask を TaskArray 化した。TaskArray 1 つに、64 個の Task が入るようにし、WordCount 対象は 166M のテキストファイルとした。TaskArray を適応した場合と、そうでない場合で比較する。SPE は 6 個使用した。time は実行にかかった時間、dma wait は、SPE が起動していた時間においての、dma 転送の待ち時間の割合。mail wait は SPE が起動していた時間においての、次の TaskList の mail を待っている時間の割合である。以下に、表にして示す。dma,mail それぞれの wait 割合に関しては spe6 個の平均を示しており、この時の Task の数は約一万個である。

表 1 WordCountTask の TaskArray 化による比較

	Task	TaskArray
time	2.184s	2.109s
dma wait	18%	12%
mail wait	5%	8%

表に示した結果より、著しい TaskArray の効果は見られなかった。この結果は WordCount においては誤差の範囲内である。効果が見られなかった原因はおそらく WordCount の場合は PPE 側の Task がないので、依存関係の解決にあまり待ち時間が発生しないからだと考えられる。また、WordCount においてはファイルをメモリにマッピングするので、ファイルの容量が大きい場合に大量にメモリを消費してしまう。その結果スワップが起きやすくなり、dma 転送の待ち時間が長くなっている可能性がある。この解決として、一度にファイル全てをマッピングするのではなく、何回かに切り分けてマッピンするのがよい。ある程度の WordCount し終わった領域に、次の WordCount 領域を入れ替えて使うことでメモリを節約でき、スワップを減らすことができるはずである。その結果メモリアクセスが高速になり、dma 転送の待ち時間も削減できる。

5.4 レンダリングの Task

レンダリングエンジンは主に、CreatePolygon、CreateSpan、DrawSpan という 3 つの Task から構成されている。それぞれの Task の動作とレンダリングの流れを示す。

1. CreatePolygonTask によって SceneGraph をもとにモデリングデータから、実際に表示するポリゴンを生成する。
2. CreateSpanTask で生成したポリゴンから Span の

生成する。

3. DrawSpanTask で Span を RGB にマッピングし描画する。ここでいう Span とは、ポリゴンに対するある特定の Y 座標に関するデータを抜き出したものである。

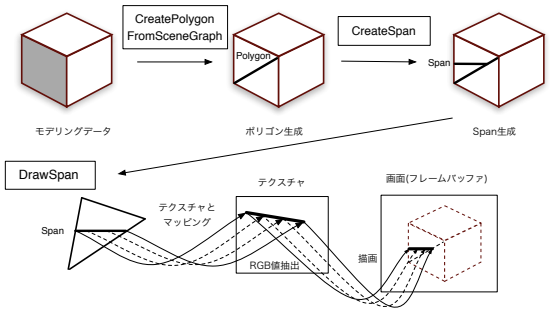

図 4　レンダリングエンジンの流れ

5.5　レンダリングエンジンの TaskArray 化

レンダリングエンジンの中で、もっとも数が多く生成される DrawSpanTask を TaskArray 化した。地球と月を表示する例題を対象に効果を検証した。FPS(Frame Per Second) は、一秒間に表示できる Frame 数のことである。

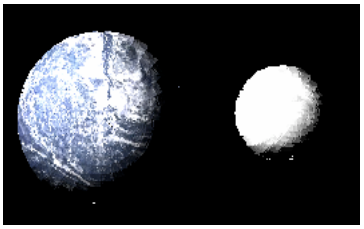

図 5　地球と月を表示する例題

表 2　レンダリング Task の TaskArray 化による比較

	Task	TaskArray
FPS	3.94	4.32
dma wait	0.06%	0.07%
mail wait	55%	42%

結果から DrawSpanTask を TaskArray 化すると、FPS が多くなり、mail の wait 時間が減ったことがわかる。レンダリングエンジンでは、PPE 側でも処理すべき Task があり、常に PPE が忙しい状態になっている。そのため mail を処理する時間が送れ SPE の mail 待ちが発生していると考えられる。TaskArray 化で Task をまとめることで SPE が 1 つの TaskList で多くの Task を実行できるようになったため、TaskList を要求する回数が減って、待ち時間が発生する回数も減少した。またそれは SPE からの mail の数が減ったということなので、PPE 側の mail 処理の時間短縮になったと考えられる。

6　まとめ

今回は Task を複数にまとめる TaskArray を提案、実装し効果を測った。TaskArray の効果があるのは、PPE 側にも実行すべき Task があり PPE が忙しい場合ということがわかった。WordCount では、PPE 側の Task がなく、mail 待ちがネックではなく、TaskArray の効果がなかった。大量のファイルをマッピングし、メモリを多く消費するのでメモリアクセス、dma 転送に待ち時間があると考えられ、TaskArray を用いてもうまく dma 転送が隠れてないようだ。dma 転送をスケジュールリングによってうまく隠す、またはメモリ領域の節約をすることができれば、今回の WordCount のような大量のデータを用いる場合の速度向上が期待できる。

6.1　メインメモリアクセス

WordCount を実装している際に極端に処理速度が遅くなる時があった。それは複数の SPE が同時にメインメモリにアクセスする際に、それぞれ離れたメモリにアクセスする時である。Task にはそれぞれアクセスすべきメインメモリのアドレスを持っており、その Task がうまく SPE に割り振られてなかったため、そのようなことがおこった。Task を SPE に割り振る際にはなるべく複数の SPE が近くのメモリをアクセスするようにした方がよい。また一定周期で SPE を同期させ、特定の SPE の Task 実行が遅くなりすぎたり、速くなりすぎたりするのを防ぐことでも、メモリアクセス先が離れることを回避できる。

参 考 文 献

[1] 宮國渡 ”Implementation of Fine-grain Task Manager for Cell” 平成 20 年度 学位論文 (修士)
[2] fixstars:http://cell.fixstars.com/ps3linux/index.php/メインページ
[3] 高山 征大「CELL REGZA」が搭載する並列化技術「Molatomium」
[4] OpenCL:http://www.khronos.org/opencl/
[5] Mark Deloura ”Game Programming Gems”